Mythen II user manual V1

July 25, 2008

Contents

1 Introduction **3**
- 1.1 Detector Performance . . . 3
- 1.2 Setting up the detector . . . 3
  - 1.2.1 Trimming . . . 4
  - 1.2.2 Setting the threshold . . . 5
- 1.3 The acquisition . . . 5
- 1.4 Post processing . . . 5
  - 1.4.1 Rate Correction . . . 5
  - 1.4.2 Flat field correction . . . 6
  - 1.4.3 Bad channels correction . . . 7

2 The hardware **8**
- 2.1 The MCS1 . . . 8
- 2.2 The MCS6 . . . 9
- 2.3 The MCS24 . . . 10
- 2.4 Changing the network configuration . . . 12

3 The software **13**
- 3.1 Installation . . . 13
- 3.2 Directory layout . . . 14
- 3.3 The virtual server . . . 15
- 3.4 The text client . . . 15

4 The GUI **19**
- 4.1 Configuration . . . 19
- 4.2 Executing the program . . . 21
- 4.3 The initialization tab . . . 22
- 4.4 The Acquisition tab . . . 26
- 4.5 The Data tab . . . 33
- 4.6 Settings . . . 35

5 Test, calibration and trimming of the detector **38**
- 5.1 Test of the modules . . . 38
- 5.2 Trimming of the detector . . . 40

5.2.1 Noise trimming 41
5.2.2 Improve the trimming using X-rays 42
5.3 Energy calibration of the detector 44
5.4 Angular calibration of the detector 46

6 Integration at the beamline **48**

7 Offline Data processing **49**
7.1 Postprocessing using root 49
7.2 Postprocessing without root 52

Chapter 1

Introduction

The MYTHEN detector is a one-dimensional microstrip detector with single photon counting readout optimized for time resolved powder diffraction experiments at the Swiss Light Source (SLS).
It allows to perform time resolved or dose critical measurements in a few seconds, instead of several minutes as with traditional crystal analyzer systems, thanks to its massively parallel detection of X-rays and fast readout.

1.1 Detector Performance

The spatial resolution is given by the strip pitch of 50 μm. For each module, 1280 channels work in parallel in single photon counting mode, thus optimizing the sensitivity of the system. The low noise of the frontend electronics allows the detection of X-rays of energy down to 4 keV, while the short shaping time of the analogue signal permits to achieve counting rates up to a few MHz/pixel. The sharp threshold dispersion allows the discrimination of the diffracted X-rays from the background of photons eventually emitted by fluorescent samples.
The dynamic range is 24 bits and the behavior is linear up to 200 kHz per channel. The detector can work up to 3 MHz per channel, even though with reduced efficiency and linearity.

1.2 Setting up the detector

Choosing the settings

In order to obtain optimal results, the detector has to be set according the needs of the application. First, the operating energy range and the maximum counting rate per channel have to be taken into account and the analog settings will then be chosen according to table 1.1.
Note that the minimum energy at which one can measure is limited by the electronic noise, while the maximum counting rate is determined by the speed of the

Settings Minimum Energy, Suggested/Maximum Rate	$\mathbf{V}_{Rgpr}$ (mV) (DACu)	$\mathbf{V}_{Rgsh1}$ (mV) (DACu)	$\mathbf{V}_{Rgsh2}$ (mV) (DACu)
High Gain 5 keV, 50/100 kHz	-126 200	-122 400	-32 260
Standard 8 keV, 100/500 kHz	-282 100	-260 300	-32 260
Fast 10 keV, 200/1000 kHz	-360 50	-399 200	30 300

Table 1.1: Detector Settings for different application requirements.

electronic. For rates higher than the suggested one, corrections must be applied to the channels in order to compensate for the loss of counting efficiency (see section 1.4 and figure 1.1). The detector will work also for counting rates larger than the indicated maximum rate, but the corrections applied will be larger and subject to larger errors. In particulary, disuniformities between channels may arise.

1.2.1 Trimming

Once chosen the analog settings, the threshold of the comparator must be chosen and set at half of the energy level in order to avoid efficiency loss and improve the spatial resolution. Moreover, the threshold needs to be selected on a channel by channel basis by means of the internal trim DACs in order to improve the count uniformity.

The trimming can be performed with or without x-rays. Trimming with x-rays is more precise but requires a uniform illumination of the detector and often long acquisition times. Trimming without X-rays gives good results for low energies (< 10 keV) but often is not good enough for higher energies and needs an improvement step. The implemented trimming methods are described in detail in section 4.3.

Together with the trimbit value for each channel, the trimming procedure selects the value of V_{trim} which is needed in order to select the most appropriate threshold for the energy used, since it defines an offset to the global threshold $Offset_{Trimming}$. This offset depends from V_{trim} and from the average trim bit value and can be calculated from the following equation:

$$Offset_{Trimming} = (aV_{trim}^2 - bV_{trim} + c) \cdot Mean(trimbits) \tag{1.1}$$

where:

$$a = \quad 5 \cdot 10^{-5}\, DAC_{threshold}/DAC_{trim}^2 \quad = 8.68 \cdot 10^{-3}\, mV^{-1} \tag{1.2}$$

$$b = \quad 0.102022\, DAC_{threshold}/DAC_{trim} \quad = 33.401653 \tag{1.3}$$

$$c = \quad 52.430851\, DAC_{threshold} \quad = 32521.4\, mV \tag{1.4}$$

Settings	**Gain** (mV/fC) (DACu/keV)	**Offset** (mV) (DAcu)	**ENC** (e^-) (keV)	$\mathbf{T}_{dead}$ (ns) (kHz at 90 %)
High Gain	207±5 15.0±0.4	487±10 804±16	195±7 0.70±0.02	650±50 150±10
Standard	160±3 (?) 11.6±0.2 (?)	491±9 (?) 817±15 (?)	230±7 (?) 0.83±0.02 (?)	170±10 (?) 590±30 (?)
Fast	125±3 9.3±0.2	497±8 829±13	262±7 0.94±0.02	90±10 1100±100

Table 1.2: Results of the calibration measurements.

1.2.2 Setting the threshold

The global comparator threshold $V_{threshold}$ is calculated according to:

$$V_{threshold} = (Offset - Offset_{Trimming}) - Gain * (Energy/2) \qquad (1.5)$$

where $Offset$ and $Gain$ are calculated from the calibration procedure and are shown in table 1.2 for the different analog settings, while $Offset_{Trimming}$ can be calculated according to equations 1.1, 1.2.

Note that the comparator must be enabled for all channels in order to be able to detect X-rays.

The procedure to calibrate the modules is more precisely described in section 5.3.

1.3 The acquisition

Once the operating parameters of the detector are set up, it is possible to start with the acquisition. The main parameter to be selected for the acquisition is the acquisition time.
In case the acquisition is gated the number of gates that should be summed up before reading out have to be specified. Moreover, the number of frames and the delay between them can be selected. In many cases some action will be performed between the frames (e.g. temperature scans etc.).
It is possible to choose how many bits of the counter will be read out, depending on the required maximum dead time between frames (still to be implemented). The method for selecting these parameters will depend on the sofware used.

1.4 Post processing

1.4.1 Rate Correction

First of all, one should correct for the loss of linearity at high rates, which causes a loss of efficiency. For high counting rates, the measured rate is smaller than

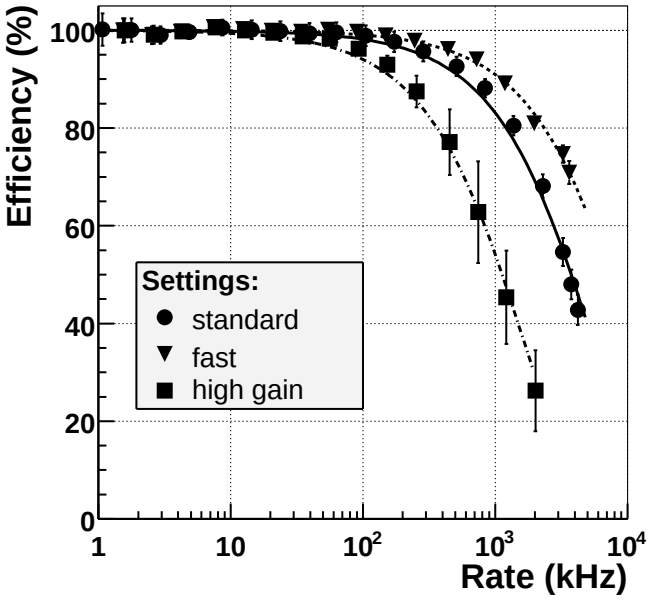

Figure 1.1: Behavior of the counting efficiency as a function of the detected photon rate for the three settings.

the rate of the X-rays absorbed in the detector n. This results in an efficiency which is dependent on the rate:

$$m = \epsilon(n) \cdot n \tag{1.6}$$

The high rate behavior can be approximated with that of a paralizable detector, thus the efficiency at rate n will be:

$$\epsilon(n) = e^{-T_{dead} \cdot n} \tag{1.7}$$

where the calibrated values of T_{dead} for the three settings are shown in table 1.2.

The rate n can usually be approximated with the measured rate m and will be calculated as:

$$n = \frac{m}{\epsilon(m)} \tag{1.8}$$

1.4.2 Flat field correction

In order to compensate for disuniformities between channels, a flat field frame (i.e. a frame with a uniform illumination of the detector and possibly at least

10E+5 counts/channel or more if required by statistics) should be acquired before or after starting the measurments at the same energy and threshold at which the measurements will be performed (one should acquire a flat field for each energy and threshold). This can usually be achieved with a capillary filled with a fluorescent substance or with a water capillary that scatters the X-rays (in this case take care of possible diffraction peaks) or by scanning the detector in front of the **attenuated** beam. If the flatfield is acquired by scanning through the beam make sure the rate is acceptible, i.e. attenuate the beam that the highes count value in 1s measurement time without scanning is inside the linear count rate region of the selected setting.

The data acquired during the measurement n_i for channel i will then be normalized with the flat field data f_i:

$$c_i = F \cdot \frac{n_i}{f_i} \tag{1.9}$$

where F is the average of the flat field values on all channels.

1.4.3 Bad channels correction

Finally, the dead or exteremely noise channels should be corrected. This can be done by completely discarding their value from the pattern acquired or by substituiting it with the average of good working neighbours.

Chapter 2

The hardware

2.1 The MCS1

For single module systems the setup consists of a detector module and a MCS1 (Mythen Control System 1) as shown in figure 2.1.
The module, closed in its box in order to protect it from the light (the cover should be taken away to allow X-rays to be detected) has to be connected to the MCS before giving power.
In order to optimize the electronics noise and help the heat dispersion, the module should be connected to the local ground through the holes in its package.

The MCS, **with the module and the ethernet already connected** can be powered up.
It's network can be configured with DHCP or with a fixed IP address. The MCS can be rebooted by pressing the reset button or by disconnetting and reconnetting the power cable.

The MCS has three LEMO connectors:

- *IN* works as input for the gate or trigger signal that can be used to synchronize the acquistion with other instrumentation. When using a trigger signal, its length should be short compared to its frequency and to the duration of the acquisition. The standard is 0-3.3 V LVTTL.
- *IN/OUT* works as output for the gate or end gate signal. It can be fed to other instruments in order to synchronize the acquisition. The standard is 0-3.3 V LVTTL.
- *HV* is for giving an external high voltage to the sensor. An internal jumper should has to be switched in order to use it.

The MCS has a serial port for debugging. The serial interface parameters are: 115200 8N1.

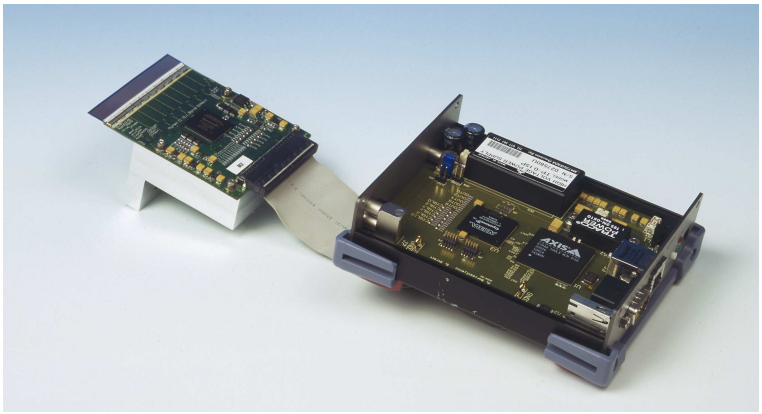

Figure 2.1: Picture of the single module setup.

The *acq active* LED will blink during the acquisition, while the *error* LED will turn on when an error occurs, usually when the frame rate required is higher than the one supported by the system.

2.2 The MCS6

For 1 to 6 modules the MCS6 should be used (Mythen Control System 6) which can control up to six detector modules as shown in figure 2.2. When the number of modules is smaller than 6, they should however be connected starting from module 0.
The modules should be closed in a box in order to protect them from the light and should be connected to the MCS before giving power.
In order to optimize the electronics noise and help the heat dispersion, the modules box should be connected to the local ground.

The MCS, **with the modules and the ethernet already connected** can powered up.
As for the MCS1 the network can be configured as DHCP or with a fixed IP address. The default is generally DHCP and you should be informed of your host name.
The MCS can be rebooted by pressing the reset button or by disconnetting and reconnetting the power cable.

The MCS6 has five LEMO connectors:

- *Gate in* works as input for the gate signal that can be used to synchronize the acquistion with other instrumentation. The standard is 0-3.3 V LVTTL.

- *Trigger in* works as input for the trigger signal that can be used to synchronize the acquistion with other instrumentation. The signal duration should be short compared to its frequency and to the duration of the acquisition. The standard is 0-3.3 V LVTTL.
- *Gate out* works as output for the internally generated gate signal i.e. it is active while the detector is counting. It can be fed to other instruments in order to synchronize the acquisition e.g. gate a radiation monitor. The standard is 0-3.3 V LVTTL.
- *Trigger out* works as output trigger i.e. gives a clock signal when the detector has finished counting one frame and is read out. It can be fed to other instruments in order to synchronize the acquisition e.g. trigger the movement of motors. The standard is 0-3.3 V LVTTL.
- *HV* is for giving and external high voltage to the sensor. An internal jumper has to be switched in order to use it.

The MCS also has a serial port for debugging. The serial interface parameters are: 115200 8N1.

The *modules* LEDs will be on when the module are accessed for sending control signals.
The *acq active* LED will blink during the acquisition, while the *error* LED will turn on when an error occurs, usually when the frame rate required is higher than the one supported by the system.
The behavior of these last two LEDs is not implemented in the current version of the board.

2.3 The MCS24

For large detectors with up to 24 modules (e.g. for powder diffraction) the MCS24 (Mythen Control System 24) as shown in figure 2.3 should be used. When the number of modules is less than 24, they should however be connected starting from the slot for module 0.
The modules should be closed in a box in order to protect them from the light and should be connected to the MCS before giving power.
In order to optimize the electronics noise and help the heat dispersion, the module box should be connected to the local ground.

The MCS, **with the modules and the ethernet already connected** can then be powered up.
As the other MCS also MCS24 can be configured with DHCP or with a fixed IP address. The default is generally DHCP configuration and you should be informed of your host name.
The MCS can be rebooted by pressing the reset button or by disconnetting and reconnetting the power cable.

The MCS24 has five LEMO connectors:

Figure 2.2: Picture of the 6 modules setup.

- *Gate in* works as input for the gate signal that can be used to synchronize the acquistion with other instrumentation. The standard is 0-3.3 V LVTTL.
- *Trigger in* works as input for the trigger signal that can be used to synchronize the acquistion with other instrumentation. The signal duration should be short compared to its repetition frequency and to the duration of the acquisition. The standard is 0-3.3 V LVTTL.
- *Gate out* works as output for the internally generated gate signal i.e. it is active while the detector is counting. It can be fed to other instruments in order to synchronize the acquisition e.g. gate a radiation monitor. The logic standard is 0-3.3 V LVTTL.
- *Trigger out* works as output trigger i.e. gives a clock signal when the detector has finished counting one frame and is read out. It can be fed to other instruments in order to synchronize the acquisition e.g. trigger the movement of motors. The logic standard is 0-3.3 V LVTTL.
- *HV* is for giving and external high voltage to the sensor. An internal jumper should be switched in order to use it.

The MCS24 has a serial port for debugging, but there is an error in the board and the pins are mirrored, so a coverter is needed in order to use it. The serial interface parameters are: 115200 8N1.

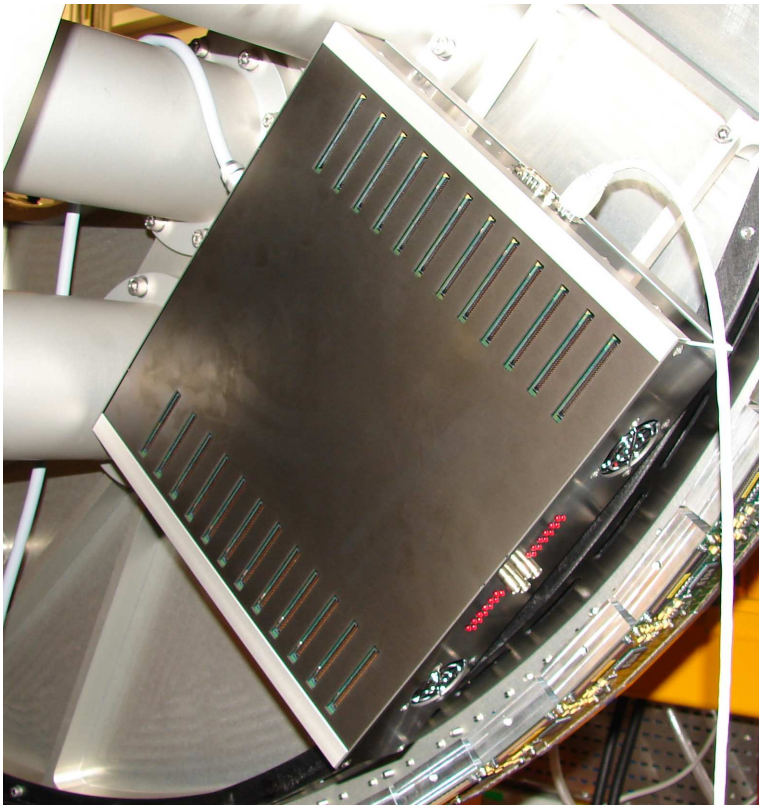

Figure 2.3: Picture of 24 modules MCS at the Material Science beamline at the SLS.

The *module* LEDs will be on when the module are accessed for sending control signals.

2.4 Changing the network configuration

If the MCS (1 to 24) is reachable through the network it can be reconfigured using its http interface (http://name or ip adress of the MCS). An other possibility is to manually change the file /etc/conf.d/net.eth0 on the MCS. This can either be done by logging in using telnet or ssh (user:root password:pass) or using the serial connection and a terminal program like minicom under linux or hyperterm under windows (especially when the MCS is not reachable through the network). The parameters of the serial connection are 115200 8N1. To change the configuration edit the file by issuing /bin/easyedit /etc/conf.d/net.eth0 .

Chapter 3

The software

The software is being developed in house and can contain bugs. please report them or ask for improvements or support at anna.bergamaschi@psi.ch or bernd.schmitt@psi.ch.

3.1 Installation

The actual software for the Mythen II system (MCS1 to MCS24) runs on Scientific Linux machines (SL3 and SL4 versions have been tested, gcc 3.2 or 3.4).

A Qt version equal or higher to 3.3 (but lower than Qt4) should be installed on the PC. It can be downloaded from
http://www.trolltech.com/developer/downloads/qt/index and the enviroment variable QTDIR should be set to its path. Qt should be compiled with the options -thread -no-xft -qt-gif -no-exceptions (check in $QTDIR/config.status)

Also a root version higher than 5.13 **with Qt support enabled** should be installed (the binaries can be downloaded from http://root.cern.ch or compile with the option --enable-qt) and the system variable ROOTSYS should be set to its path. Further details can be found in the root user's manual at http://root.cern.ch/root/doc/RootDoc.html in chapter 1 (Introduction), 26 (ROOT/Qt integration interfaces) and 28 (Install and Build ROOT).

ROOTSYS/bin and QTDIR/bin should be added to the PATH;
ROOTSYS/lib and QTDIR/lib should be added to the LD_LIBRARY_PATH [1].

Finally you should define the enviroment variable MYTHENDIR addressing to the directory where you will install the acquisition software and add MYTHENDIR/bin to your PATH.

[1] In some linux installation there might be configuration scripts e.g in the directory /etc/profile.d/ and /etc/ld.so.conf.d/ overwriting this variables, so check that the installations correspond. If you have already used other Qt or Root versions, it might be that the old libraries are loaded in cache. To delete the chache, remove the file /etc/ld.so.cache and run .....

As an example, we suggest to add the following lines to your `.bashrc` or `.profile` file (changing `QTDIR` and `ROOTSYS` accordingly):

```
export ROOTSYS=/local/root
export QTDIR=/local/qt
export MYTHENDIR=/local/NewMythenMCS

export PATH=$QTDIR/bin:$ROOTSYS/bin:$MYTHENDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTDIR/lib:$ROOTSYS/lib:$LD_LIBRARY_PATH
export MANPATH=$QTDIR/doc/man/usr/local/man:$MANPATH
```

You should now be able to install the acquisition software by copying the main directory NewMythenMCS.

You can compile the software by simply executing:

```
$ make clean
$ make
```

If the compilation succeeds, the setup of the system should be fine.

3.2 Directory layout

The directory layout is as follows:

NewMythenMCS is the main directory

NewMythenMCS/bin directory containing the executables:

- **NewMythenII** the GUI (see chapter 4
- **mcs_large_client** the text client (see section 3.4)
- **mcs_stop_server_virtual, mcs_large_server_virtual** the virtual server (see section 3.3)

NewMythenMCS/docs directory with documentation (e.g. this document)

NewMythenMCS/src contains the source files of the acquisition program

NewMythenMCS/include contains the include files of the acquisition program

NewMythenMCS/objs contains the object files creted during the compilation

NewMythenMCS/scripts scripts for header file generation and integration into the beamline etc.

NewMythenMCS/calibration standard location for calibration files and for the bad channel list

NewMythenMCS/root directory with root scripts for energy calibration, post-processing etc

NewMythenMCS/trimbits standard location for default trim files (trimbits all set to 0) for standard, highgain and fast settings

NewMythenMCS/trimbits/beamline/highgain standard location for highgain settings trim files (will be loaded automatically when highgain settings are selected)

NewMythenMCS/trimbits/beamline/standard standard location for standard settings trim files (same as above for standard settings)

NewMythenMCS/trimbits/beamline/fast standard location for fast settings trim files (same as above for fas settings)

NewMythenMCS/data possible data directory. Can be linked to your fileserver

NewMythenMCS/config contains the files necessary for configuring the GUI (see section 4.1)

NewMythenMCS/angcal contains the angular calibration

3.3 The virtual server

The virtual server has been developed in order to be able to test the acquition program functionality.
Just execute:

```
$ mcs_stop_server_virtual &
$ mcs_large_server_virtual &
```

in a terminal and every time you will execute some command from the GUI or the text client it will print a message concerning the commands it has received (assuming that you have configured the localhost as your acquition board). mcs_stop_server_virtual simulates the emergency stop server, while mcs_large_server_virtual simulates the acquisition program.
The virtual servers have not been debugged in detail, so it might be that it gives some error even if the command received are correct.
It can also be useful in order to check the reason of possible communication problems to the MCS (e.g. activated firewalls).

3.4 The text client

The text client is useful in order to integrate the detector in an experimental setup controlled via command line. It is usually convenient to initialize the detector and setup the measurements using the GUI and then acquire the data or change some small parameters (e.g. the acquisition time) by using the text client. Moreover the text client does not post process the acquired data (rate and flat field correction, angle conversion, data merging) as it is authomatically

done by the GUI. These operations should then be performed offline by the user using the postprocessing programs provided.
Also the text client has been debugged in detail only for what concerns the data acquisition and not for more complex functions (e.g. trimming) which are usually performed by using the GUI.

To run the text client execute:

```
$ mcs_large_client myhost [args]
$
```

where `myhost` is the host name or IP address of the MCS (or of your PC in case you are using the virtual server) and `args` is and argument listed in this list which allows you to control the detector.

-help prints this help

-start starts the acquisition with the current settings

-stop stops the acquisition

-readout reads out the current content of the counters

-time duration of the acquisition in ns

-delbef delay between trigger and frame in ns

-delafter delay between framesin ns

-frames number of consecutive frames

-gates number of gates in gated acquisition

-fname selects the file to write the data to

-startindex selects the startinf index of file to write the data to (if not specified everything is written on one file

-clkdiv clock divider (with respect to 100 MHz)

-inpol if 0 input active low, if 1 active high

-outpol if 0 output active high, if 1 low

-gateen 1 enable gating 0 disables

-trigen 1 enable single trigger 0 disables

-conttrigen 1 enable continous trigger 0 disables

-all selects all channel/chips/modules

-chan selects the channel

-chip selects the chip

-mod selects the module

-load loads a trim file

-trimfile selects the trimfile to load/where to save the new trims

-vthresh sets the threshold

-vtrim sets vtrim

-vcal sets vcal

-rgsh1 sets rgsh1

-rgsh2 sets rgsh2

-rgpr sets rgpr

-fifo selects the fifo

-trim sets the trim value for the selected channels

-coe enables/disable the comparator for the selected channels

-counts sets the counter value for the selected channels

-outmux selects the output mux mode

-trimnoise trims using the noise algorithm

-trimbeam trims using the beam algorithm

-trimimprove improves the trimming

-trimfixed improves the trimming

-trimfile selects the trimfile to load/where to save the new trims

-nsigma resolution of the trimming (4 high, 2 low)

-countlim average counts for trimming

-cal calibrates the detector

-calsens calibrates the detector through the sensor

-fname selects the file to write the data to

-ncal umber of calibration pulses

-digitest tests the module

-fpgatest tests the FPGA

-get get a variable value (can be: time delbef, delafter, frames, gates,gateen, trigen, conttrigen, clkdiv, inpol, outpol, vthresh, vtrim, vcal, rgsh1, rgsh2, rgpr, trim, modnum)

-fname selects the file to write the data to

To start the acquitition and readout the detector one will normally use:

```
$ mcs_large_client myhost -start -fname myfile.raw
$
```

Chapter 4

The GUI

4.1 Configuration

Before running the acquisition program you should configure your system. The default settings are defined in the file src/config.txt. You can change the name of this file or use alternative files, but in this case you will have to call them as options when executing the acquisition program. The following variables should be defined, or the default values will be used:

HOST is the host name or IP address of the MCS (see section 2.4). The default is *mcs01*

NMOD is the number of detector modules installed on the MCS. It can be any number smalled than the maximum number of modules hosted by the MCS (1 for MCS1, 6 for MCS6, 24 for MCS). If NMOD is smaller then the actual number of installed modules, only the first NMOD will be readout i.e. the maximum frame rate will be higher, but the data relative to the last modules will be lost. The default is *1*.

PROTOCOL is the data transfer protocol. It can be *udp* or *tcp*, but *udp* can be used only for single module systems. The default is *tcp*.

DATADIRLOCAL is the default directory where the data will be written. The default is *$MYTHENDIR/data*.

TRIMDIRLOCAL is the default directory where the trimbit files will be saved to and loaded from. The default is *$MYTHENDIR/trimbits*.

BADCHANNELLIST is the location of the list of the bad channels of the detector. In case you are executing the program with the option `-x04sa` (i.e. synchrotron beamline options) the deafult will be *$MYTHENDIR/calibration/badchannel_detector.list*, otherwise the software will serch for files called *$MYTHENDIR/calibration/badchannel.snxxx* where *xxx* is the serial number of the modules of your system. the bad channels list syntax should be in a format with a number corresponding

to the bad channel for each line, or in case of gropus or bad channels a line with the number of the first and that of the last bad channel separated by a -. For example:

```
10
335-338
```

would mean that channel 10 is bad and all the channels between 335 and 338 (extermes include) are also bad. Please remeber that the channel enumeration starts from 0.

TRIMFILE_STANDARD, TRIMFILE_FAST, TRIMFILE_HIGHGAIN is the location of the default trimfiles (i.e. with correct shaper settings and all trimbits set to 0. See section 4.6) for the standard, fast and high gain settings respectively. The default is *$MYTHENDIR/trimbits/standard.trim*, *$MYTHENDIR/trimbits/fast.trim* and *$MYTHENDIR/trimbits/highgain.trim*.

CAL_TRIMFILE_STANDARD, CAL_TRIMFILE_FAST, CAL_TRIMFILE_HIGHGAIN is the location of the detector-specific trimfiles (i.e. with correct shaper settings and trimbits, for which and energy calibration has been performed. It will usually be the noise trimmed files. See sections 4.6 and 1.2.1) for the standard, fast and high gain settings respectively. The default is *$MYTHENDIR/trimbits/beamline/standard/noise*, *$MYTHENDIR/trimbits/beamline/fast/noise* and *$MYTHENDIR/trimbits/beamline/highgain/noise* and the trimfile name is *noise.snxxx* i.e. the extension of the files will be given by the installed modules serial number.

CAL_STANDARD, CAL_FAST, CAL_HIGHGAIN is the location of the detector specific energy calibration files (see sections 4.6, 1.2.2 and 5.3). The default is *$MYTHENDIR/calibration/standard*, *$MYTHENDIR/calibration/fast*, *$MYTHENDIR/calibration/highgain* the calibration file name is *settings.snxxx* i.e. the root is the settings name and the extension of the files will be given by the installed modules serial number.

HEADER_BEFORE_SCRIPT, HEADER_AFTER_SCRIPT, MOVEDETECTOR_SCRIPT are the default script to be executed before and after each frame to write the headers and between frames to move the detector when executing the program with the option `-x04sa` (i.e. synchrotron beamline options), as explained in section 4.4 and chapter 6. The default are *$MYTHENDIR/scripts/header_before.awk*, *$MYTHENDIR/scripts/header_after.awk* and*$MYTHENDIR/scripts/move_detector.awk*.

ANGCALFILE is the location of the file necessary for the angular calibration for converting the channel values in angle when executing the program with the option `-x04sa` (i.e. synchrotron beamline options), as explained in chapter 6. It will be given by a file with a line per module with the format:

```
module 5 center 640 +- 0 conversion 6.5629E-05 +- 6.93E-09 offset 14.99771 +- 0.00015
```

where conversion coefficient and offset are determined by a calibration procedure as explained in section 5.4. The default is *$MYTHENDIR/angcal/ang.off*

BLOFFSET is the constant beamline angular offset for converting the channel values in angle when executing the program with the option `-x04sa` (i.e. synchrotron beamline options), as explained in chapter 6. The default is 1.088

DIRECTION should be set to *1* (default) if small module index corresponds to lower angles, *-1* otherwise. In this case the GUI will authomatically write the data output files with the order of the channels inverted.

EPICSPUT, EPICSGET are the epics commands to set and read the variables. They are used only for beamline integration for converting the channel values in angle when executing the program with the option `-x04sa` (i.e. synchrotron beamline options) and chapter 6. The default are *caput* and *caget*

DETECTORPOSITION_CHANNEL, ICCOUNTS_CHANNEL are the epics channels relative to the detector motor position and to the readout of the ionization chamber. They are used only for beamline integration for converting the channel values in angle when executing the program with the option `-x04sa` (i.e. synchrotron beamline options) and chapter 6 in order to authomatically convert the channel values in angle and possibly normalize for the beam intensity (not yet implemented). The default are *caput* and *caget*

DIRREMOTE is the directory on the MCS where it is possible to write the data (unused)

TRIMDIRREMOTE is teh directory on the MCS where it is possible to save and load trimbit files (unused)

4.2 Executing the program

Just power up the MCS, wait a few minutes and then execute the acquisition program from the main directory NewMythenMCS :

```
$ ./NewMythenII [args]
```

After printing some messages, a window like the one showed in figure 4.1 should open

The program can be executed with the following arguments:

-help prints the possible options for the program

-f myconf.txt to define the configuration file to myconf.txt instead of the default *config/config.txt*

-x04sa to enable the actions generally needed for the operation at the Material Science bemline at the SLS (header before and after the frames, angle conversion, data merging etc. - see chapter 6)

-size n sets the size of the text to n (the default is n=10)

-scale s scales the size of the text and the root canvas by the scaling factor s (the default is s=1). It is useful when executing the program on a PC with low screen resolution (e.g. a laptop) and hte window would then fall out of the screen.

Here are some common error messages:

```
hostname: RPC: Remote system error - Connection refused
Failed opening hostname
```

means that either you should wait a little longer until the server program starts on the MCS, your system configuration is wrong (check `HOST` in the used configuration file), the MCS is not online or the firewalls of your system prevent you from making RPC calls to the MCS (you should then check with your system managers).

```
Cannot register service: RPC: Unable to receive;
errno = Connection refused unable to register (QTSERVER, ONE, udp).
```

means that no portmapper is running on your PC. You should then install it on your system.
This is not essential for controlling the detector, but can be useful if you want to control the GUI via a local text client.

Sometimes the program might take very long to startup. This usually happens when you are using the `afs` Qt or root libraries infact in this case the program will try to connect to `afs` at startup and it will take very long if it fails (the timeout is 10 minutes!).

At startup the program will load the standard settings of the detector and set the threshold at 6 keV (good for measuring with 12 keV X-rays).

In case your detector has not been calibrated yet, you will get a lot of warnings. This only means that the energy calibration of your detector will not be accurate at this point and you have to calibrate it as described in section 5.3.

The default acquisition time will be set to 1 s.

4.3 The initialization tab

The *Initialization* tab of the GUI is used in order to initialize the detector and change its settings.

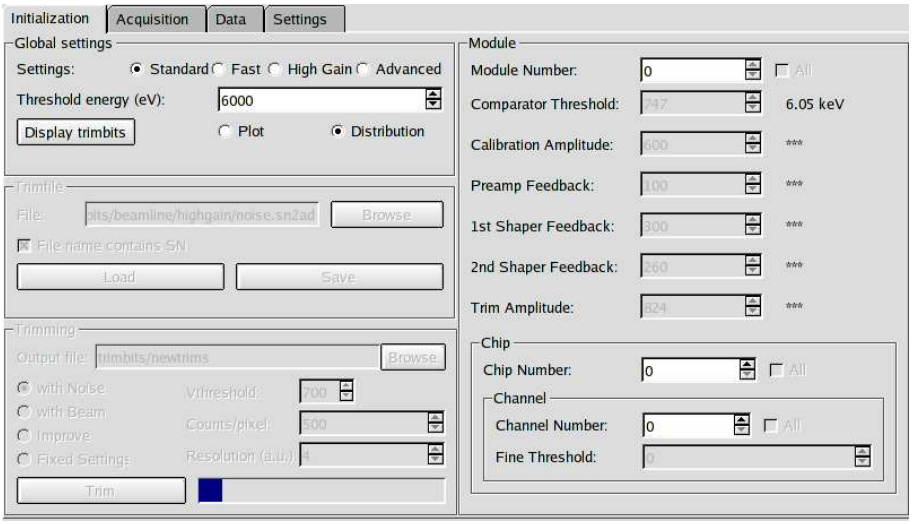

Figure 4.1: Screenshot of the *Initialization* tab (without root canvas).

The GUI will turn on in standard user mode, i.e. many options will be shadowed and you will not be allowed to perform some operations. In order to switch to expert mode, you should click on the *Advanced settings* radiobutton in the *Global Settings* box.

Global Settings allows you to change the settings of all the modules of your detector.These should be chosen as described in section 4.6 and suggested in table 1.1. When you change the settings, the corresponding trimfiles and calibration constant will be loaded, if available, otherwise, default untrimmed settings and calibration constants will be used in the acquisition.

The *Advanced* radiobutton allows you to switch to the expert mode of the GUI and to change the settings individually for each module, channel or chip. No trimfile will be automatically loaded, and no energy calibration will be possible with unknown settings. For this reason the threshold itself should be changed in the *Module* box.

The threshold energy should be set at half of the working energy (unless other needs occur in case of fluorescent samples or very low energies) and should be specified in eV. The threshold energy will be precise only in case the detector has been trimmed and calibrated (some hundreds eV resolution).

It is possible to display the actual trimbits loaded to the detector. If the trimfiles have been loaded already in this session, it will be enough to click on the radiobuttons to display their map or distribution, otherwise they have to be read from the detector by pushing the button *Display trimbits*.

Trimfile is active only in expert mode. It allows the user to load a specific trimfile to the modules. If the checkbox *filename contains SN* is not checked, the same trimfile will be loaded to all modules when pushing the *load* button and only one trimfile, corresponding to the settings of the selected module, will be saved when pushing *save*.
In case *filename contains SN* is checked, the extension corresponding to the modules serial number will be added to the file name and a different file will be loaded (or saved) individually for each module.

Trimming is active only in expert mode and allows to trim the detector (see section 1.2.1). The trimming is started by pressing *trim*

- *Output file* is the file where the trim values will be saved. Since all modules will be trimmed, the extension relative to the modules serial number .snxxx will be appended to the file name.
- The radio button list allows to choose the *trimming mode*.
  The acquisition time for trimming should be set in the *Acquisition* tab
  - *with Noise* means that the trimbits values are chosen so that all channels have the same number of noise counts at a certain threshold. The trimming procedure must be performed **without X-rays**.
    In this case *Vthreshold* should usually be about 700 DACu (can be changed if the trimming is not satisfactory) and *Counts/pixel* usually between 50 and 2000 in 100 ms (time set in the *Acquisition* tab). Noise trimming is described in more detail in 'section5.2.1.
  - *with Beam* means that the trimbits values are chosen so that all channels have the same number of counts at a certain threshold. The trimming procedure must be performed wit X-rays. In this case *Vthreshold* should be set at a value where the first channels start counting about half of the X-rays. *Counts/pixel* is the number of counts (on the threshold scan plateau) in the time window set in the *Acquisition* tab. The best results are obtained with X-rays at half of the energy that will be then used during the measurements.
    It is important that the illumination is flat!. This method has not been largely tested.
  - *Improve* is used when the detector is already trimmed (e.g. with noise or at other energies) and the trimming needs to be improved. The procedure should be much faster than completely trimming with beam since only few trimbits values are scanned. However when the system is composed by many modules it can be even slower than the *beam* method since initializing all channels individually can be slower than the acquisition itself (except for very low rates).

In this case *Vthreshold*, *Counts/pixel* and *Resolution*) are unused, but the threshold of the system should be set by using the *Threshold energy* spinbox or the *Module* setting in the initialization tab.
Remember to select the duration of the time window in the *Acquisition* tab. **It is important that the illumination is flat!**.
This method has not been largely tested yet.

- *Fixed settings* is performed without automatically selecting *Vthreshold* and *Vtrim*. This is particularly useful when there are dead chips or modules that would not allow a global trimming.
Vthreshold and *Vtrim* should also be set in the *Initialization* tab.
Counts/pixel has to be set at the level where all channels should be trimmed (half of the counts on the plateau in the threshold scan). *Resolution* is not used.
When using X-rays, it is important that the illumination is flat, but it works also with noise!

- *Vthreshold* is the DAC value at the beginning of the trimming. It should be set according to the trimming mode (used only in *noise* and *beam* mode).
- *Counts/pixel* is needed in order to define the right trimming parameters and should be set according to the trimming mode (used in *noise*, *beam* and *fixed settings* mode).
- *Resolution* defines how good one wants to trim. The lower it is, the larger will be the trimbits distribution i.e. the trimming will exploit a smaller dynamic range with a higher resolution (smaller threshold correction per bit) and will thus be more precise, but more channels will be outside the range and therefore wrong. It should usually be set between 3 and 5. Used only in *noise* and *beam* mode.

The Module, Chip and Channel boxes are always active for checking the settings of the various modules and channels, but the expert mode has to be enable in order to modify them. In order to update the view of the settings, just toggle the *Module*, *Chip* and *Channel* spinboxes
The values are changed only for the selected module, chip and channel unless the *All* checkbox are checked. Al three checkboxes should be checked so that the changes are made to all modules, chips and channels.
By changing the value of the *Module* spin box, the settings relative to the selected module, chip and channel will be displayed.

- *Comparator threshold* is the comparator theshold $V_{threshold}$ in DAC units. When known settings are used, the label is updated to the corresponding energy values. The values are more close to reality if the trimbits have been plotted.
- *Calibration amplitude* is the amplitude of the calibration pulse $Vcal$ that can be applied to the chip calibration input or to the sensor backplane. It has not been calibrated yet.

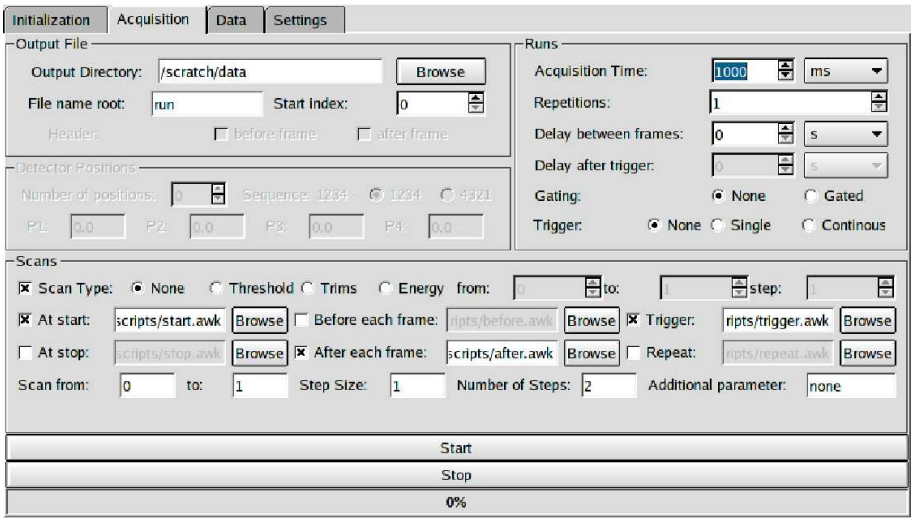

Figure 4.2: Screenshot of the *Acquisition* tab (without root canvas).

- *Preamp feedback* is the value of R_{gpr} in DAC units.
- *1st shaper feedback* is the value of R_{gsh1} in DAC units.
- *2nd shaper feedback* is the value of R_{gsh2} in DAC units.
- *Trim amplitude* is the value if $_{Vtrim}$ in DAC units. The label is updated to the average value in threshold DAC units to which each trimbit corresponds.

By changing the value of the *Chip* and of the *Channel* spin box, the *Fine threshold* (trimbit value) relative to the selected channel of the selected chip will be displayed.

4.4 The Acquisition tab

The *Acquisition* tab of the GUI is used in order to set all the important parameters of the acquisition with the Mythen detector. Some options will be shadowed unless the GUI is executed with the option `-x04sa` or the expert mode is enabled by clicking on the *Advanced* radiobutton in the *Global Settings* box in the *Initializaion* tab. The sequence with which the different scans will be executed is shown in figure 4.7.

The *Output file* box allows to select where the data will be saved:

- *Output directory* is the directory where the files will be written. It can be browsed with the *browse* button. If the directory does not exist, it will be created down to one level.

Script scan *v*	**Advanced scan** *advanced* mode *s*	**Position scan** *x04sa* mode *ip*	**File name**
no	no	no	data/run_15.dat
yes	no	no	data/run_scan1.253000_15.dat
no	yes	no	data/run_adv500_15.dat
no	no	yes	data/run_p2_15.dat
yes	yes	no	data/run_scan1.253000_adv500_15.dat
yes	no	yes	data/run_p2_scan1.253000_15.dat
no	yes	yes	data/run_p2_adv500_15.dat
yes	yes	yes	data/run_p2_scan1.253000_adv500_15.dat

Table 4.1: Format of the file names where the *output directory* is `data`, the *root file name* is `run`, the *script scan variable* v is 1.253, the *advanced scan variable* a is 500, the *position index* ip is 2 and the *run number* i is 15.

- *File name root* is the beginning of the file name. The complete file name will be created according to the format described in table 4.1 according to the options enabled in the acquisition.
- *Start index* is the index attached to the file name in order to distinguish between different runs. In case of multiple runs, it will automatically be incremented. For the first run, if the file with the selected index already exists, the index will be incremented until unique. In figure 4.7 it is indicated as *i0*.
- the *Header* option is enabled only when the GUI is executed with the `-x04sa` flag.
  When *before frame* is checked, the system command:
  `$ header_before nrun=i fn=fname`
  is executed before each frame, while when *after frame* is checked, the system command:
  `$ header_after nrun=i fn=fname`
  is executed after each frame (see figure 4.7), where i is the run number and fname is the filename where the data of the corresponding frame will be saved.
  The scripts `header_before` and `header_after` can be defined in the configuration file (see section 4.1). The default scripts used at the SLS place EPICS calls and write the beamline variables in an header file *fname.parab*. These AWK scrip can be easly edited for every need.

The *Runs* box allows to select all the useful parameters for the acquisition:

- *Acquisition time* corresponds to the acquisition time in normal mode (see figure 4.3) and to the number of gates in gated mode (see figure 4.4). It is

updated everytime the value is changed, so it is recommended always to modify the value and check the unit after changing the acquisition mode.

- *Repetitons* is the number of measurements acquired (*nrun* in figure 4.7). When no scan or header option is required the acquisition is carried on in real time and might time out for short acquisition times due to the time needed to transfer the data.

- *Delay between frames* is the time between to consecutive frames (see figures 4.3 and 4.5). the minimum delay time is the readout time, which is by default of the order of 300 μs. This does not make sense if header acquisition, detector movement or scan options are enabled since the delay between frames is limited by these options and the consecutive frames are not acquired in real time.

- *Delay after trigger* is used only in trigger mode and represents a waiting time between the external trigger and the beginning of the acquisition (see figures 4.5 and 4.6).

- *Gating* allows to enable the *Gated* option i.e. the detector is counting only when the external gate is active, as shown in figure 4.4.
  The number of gates in each frame before read out is given by the *Acquisition time* spinbox, while the number of frames is given by the *Repetitions* spinbox.
  The polarity of the gate signal can be selected in the *Advanced* tab.

- *Trigger* allows to synchronize the acquisition with an external trigger signal. The edge on which the acquisition starts can be selected in the *Advanced* tab.

  - The *single trigger* mode is illustrated in figure 4.5: the entire number of repetitions is executed in real time after one trigger. For this reason it makes no sense to choose this mode when header and scan options are enabled.
  - The *continous trigger* mode is illustrated in figure 4.6: for each repetition one trigger signal is needed. If the header or scan options are enabled, the triggers will only be accepted when all scripts have been executed i.e. some triggers might be lost.

The *Detector position* box is enabled only when the GUI is executed with the `-x04sa` flag. The whole position scan is executed for each value of the other possible scans chosen.

- *Number of positions* allows to chose in how many positions the detector will be moved in order to acquire each measurement (in figure 4.7 indicated with *np*). If 0, the detector will not be moved by the acquisition program, otherwise the detector will be moved to the positions specified

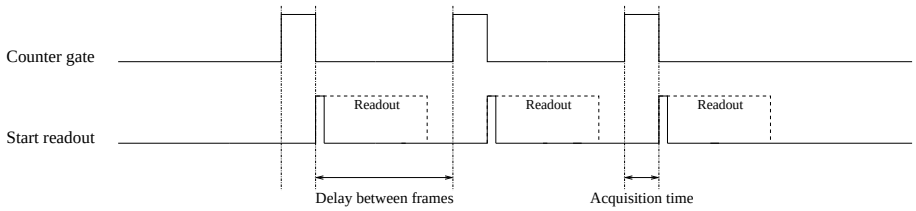

Figure 4.3: Scheme of the normal acquisition (no gate and no trigger) with *Number of repetitions* set to 3. The arrows show the meaning of the *Acquisition time* and *Delay between frames*. If this last parameter is shorter than the acquisition time it will be ignored and the following frame will start just at the end of the readout process.

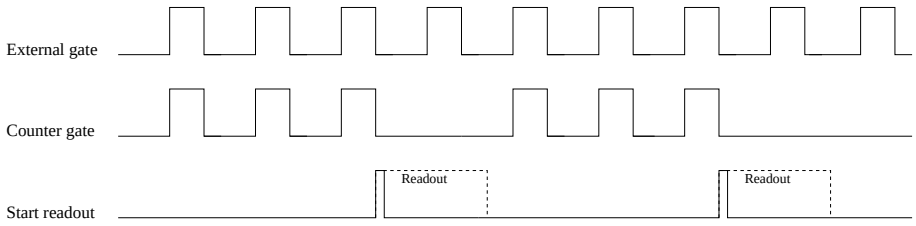

Figure 4.4: Scheme of the gated acquisition with acquisition time set to 3 gates and number of repetitions set to 2. All signals are active high.

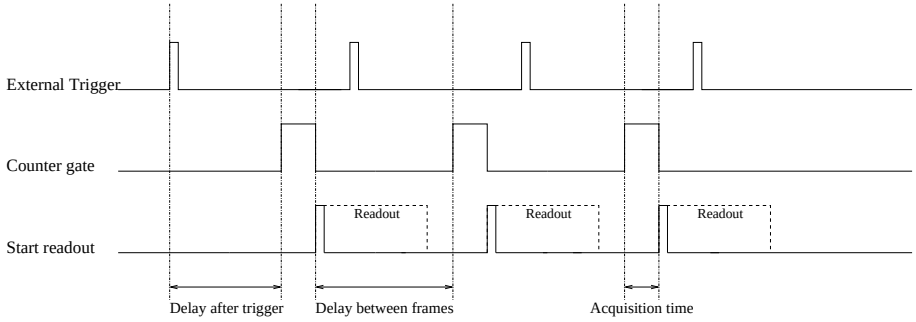

Figure 4.5: Scheme of the single trigger acquisition mode with number of repetitons set to 3. The frames are acquired in sequence after the first trigger, ignoring the following. The arrows show the meaning of the *Acquisition time*, *Delay between frames* and *Delay after trigger* parameters. The trigger is configured to start on the rising edge and all signals are active high.

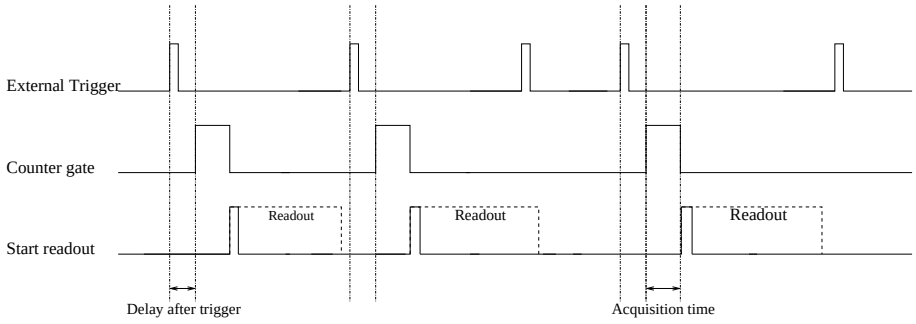

Figure 4.6: Scheme of the continous trigger acquisition mode with number of repetitons set to 3. A frame is acquired after each of the 3 trigger signals. The arrows show the meaning of the *Acquisition time* and *Delay after trigger* parameters. The trigger is configured to start on the rising edge and all signals are active high.

in the entries *P1*, *P2*, *P3*, and *P4* by calling before each frame the system command: `$movedetector pos=Px`
`$`
where movedetector is the detector movement script configured in the confguration file (see section **??**) and Px is the correspondent position.

- *Sequence* allows to choose wether the detctor is moved between the positions always in the same sequence (1234-1234) or if the sequence of the positions will be inverted (1234-4321) in order to save time.

The *Scans* box can be enabled by simply clicking on the relative checkbox. Unless in expert mode, the first line of scan modes (*advanced scan*) will be shadowed.

- When *Start script* is checked, the script specified in the relative entry is executed at the beginning of the whole series of measurments (see figure 4.7) with argument `nrun=i par=s` where i is the run number and s is the string in the *Additional parameter* text entry. It can be useful e.g to open the shutter at the beginning of the series of measurements.

- When *Stop script* is checked, the script specified in the relative entry is executed at the end of the whole series of measurments (see figure 4.7) with argument `nrun=i par=s` where i is the run number and s is the string in the *Additional parameter* text entry. It can be useful e.g to close the shutter at the end of the series of measurements.

- When *Script before frame* is checked, the script specified in the relative entry is executed before the measurment (see figure 4.7) with argument `nrun=i var=v par=s` where irun is the run number, v is the value of the

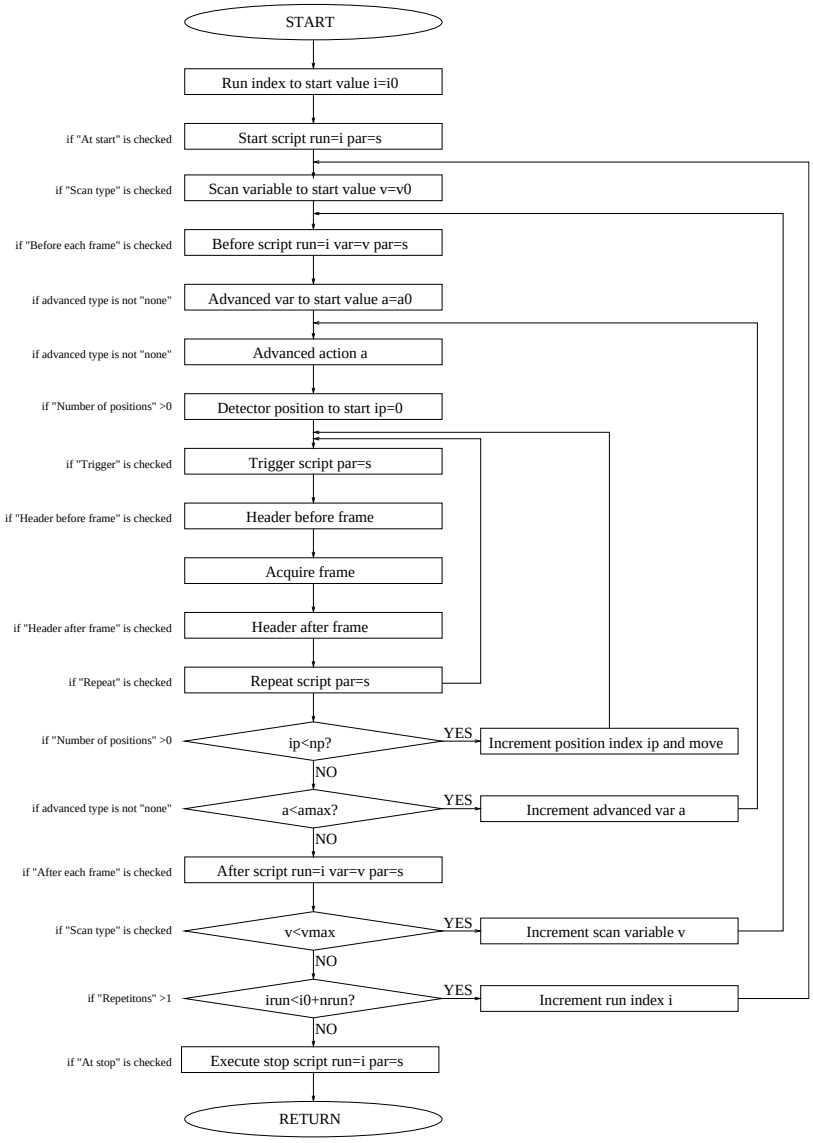

Figure 4.7: Sequence of the acquisition when the various scan options are enabled. v is the scan variable, a the advanced scan variable, i the run index, ip the position index and s the additional string parameter.

scanned value and s is the string in the *Additional parameter* text entry. It can be useful e.g to change a temperature.

- *W*hen *Script after frame* is checked, the script specified in the relative entry is executed before the measurment (see figure 4.7) with argument `nrun=i var=v par=s` where i is the run number, v is the value of the scanned value and s is the string in the *Additional parameter* text entry. It can be useful e.g to change a temperature.

- *W*hen *Trigger* is checked, the trigger script will be executed before each frame (see figure 4.7) with the argument `par=s` and the acquisition will start only when it gives a return value 0 (with 1 it will keep on executing the trigger script every 500 ms until the condition contained in there is not met or the stop button is pressed). It might be useful for example to check that there is beam (read the IC or the ring current and return 0 if they are >0, 1 otherwise).

- *W*hen *Repeat* is checked, the repeat script will be executed at the end of each frame (see figure 4.7) with the argument `par=s` and the acquisition will continue only when it gives a return value 0 (with 1 it will keep on repeating the measurements until the condition is not met or the stop button is pressed). It might be useful for example to repeat the acquisition if there has been a beam dump (read the IC or the ring current and return 0 if they are >0, 1 otherwise). If coupled to the same trigger script, in case of a beam dump the last frame will be repeated and started again only when the beam is back again.

- The last line of the box allows to define the *scan variable range* i.e the start *v0* and stop *vmax* value and the step size, which can be a floating point number, or the number of steps, which should be an integer. It is also possible to define an additional string parameter *s* (it might also be a number) which is then passed to the scripts. It could be for example the name of a motor, so it would be possible to use the same motor movement script for all motors and then pass the name of the motor as a parameter. The value of the entries is update when by pressing the enter key.

- The first line of the box is active only when in expert mode (click *Advanced* int the *Global settings* box in the *Initialization* tab) and allows to define the *advanced scan* mode and range.

  - *Threshold* changes the value of the comparator threshold for all modules. Useful for selecting the right threshold and calibrating the detector (see section 5.3). The advanced variable range can vary between 200 and 900 and is expressed in DAC units.
  - *Trims* changes the value of the trimbits. Useful to perform offline trimming. The advanced variable range can vary between 0 and 63.
  - *Energy* not yet implemented. It will scan the module threshold so that each step will correspond to the same energy for all modules.

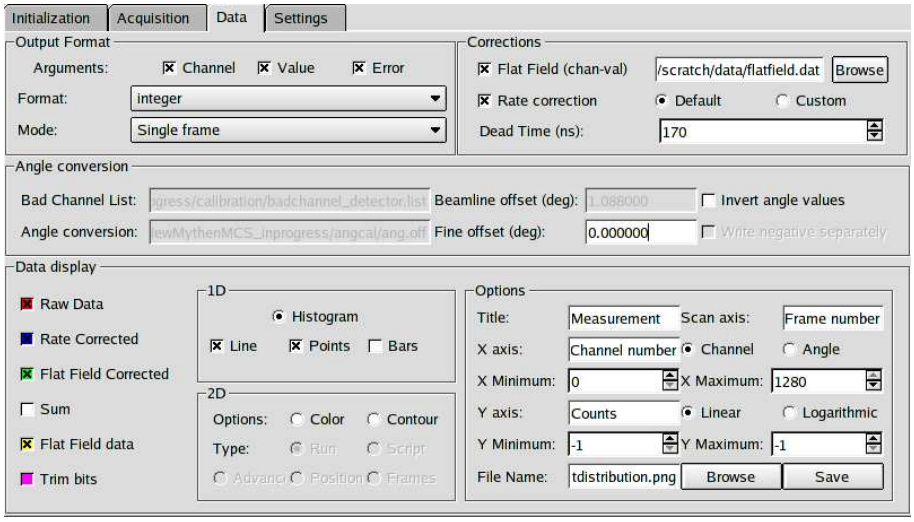

Figure 4.8: Screenshot of the *Data* tab.

- *Command* executes a system command between frames. It can also be a system script or an empty string in order to allow acquisition for short time without timeouts. If the command contains the string %i or %d, it will be substituted with the value given by the scan range.
- The *Advanced variable range and step size* can be defined by using the realtive spin boxes and the advanced variable can then assume only integer values (which is however enough for threshold expressed in DAC units, trimbits and X-ray energy expressed in eV).

4.5 The Data tab

The *Data* tab allows to select the format of the data output, the type of postrocessing and the style of the data plots.

The *Output format* box allows to select the format of the output files. A file with the extension *.raw* containing two columns corresponding to the channel number and detector readout is always generated for each frame. Additional file with ,ore arguments or postprocessed data with the extension .dat will be generated depending on the options selected.

- *Arguments* selects the data to write in the output file with the extension *.dat*. It is possible to check angle (only with the *-x04sa* option), channel, value and error. If only channel and value are checked no *.dat* file will be generated for the raw data, since the data are already present in the *.raw* file.

If *Angle* is selected, the angle conversion is performed using the angular calibration file and beamline offset specified in the config file (see section **??**) and the fine offset specified in the *angle conversion* box in this tab. I case of data acquired in several positions, the data will be merged to obtain a single data file called "merged".

- *Format* selects the format how the values and errors are written in the output file. The default is integer but also float or hexadecimal can be chosen.
- *Mode* selects how the data will be written.
  * *Single file* is the default and one file per frame is written.
  * *Image* all frames are saved in one file, one frame per row (only values, without angle, channel number or erros).
  * *Sum frames* one file with data summing up all frames is written.
  * *No file* is written.

The *Corrections* box allows to perform the rate and flat field corrections on the raw data (see section 1.4). Performing the corrections online, makes the acquisition slower.

- *Flat field* when checked the flat field correction is automatically performed and the corrected data are saved with the extension .ff. A frame with a high statistics and a flat illumination at the same energy used for the acquisition should be selected in order to calculate the flat field coefficients. When the flat field file is selected the calculated correction coefficients for the channels will be plotted. The channels counting 0 in the flat field file will be discarded and considered as bad channels.
- *Rate correction* when checked, the rate correction is automatically performed (before the flat field correction, if both are checked) and the corrected data are saved with the extension .rc. It is possible to use the *default* dead time, calibrated from the settings selected in the *Detector* tab, or to chose a *custom* one, which should be indicated in ns.

The *Angle conversion* box is used only with the *-x04sa* option and box and shows the parameters used for the angle conversion, as set in the configuration file (see section **??**). It is also possible to set a fine angular offset which can be then used for the angular conversion. It is also possible to click *Invert angle values* in order to obtain positive values when the detector is positioned at negative angles. Clicking on *Save negative separately* two files corresponding to the positive and negative angular range (inverted) will be produced in case the detector is placed in an intermediate position (not yet implemented).

The *Data display* box allows to select the data to be displayed and in which format.

- *Argument* allows to select the data to display, after an acquisition (the trimbits have to be plotted from the *Advanced* or the *Detector* tab in order to be displayed). The colour of the check box represent the colour of the data in the 1D plot.
- *Plot type* When 1D histogram is selected, the last frame (when available) is plotted in an histogram with the graphic options selected (*Line, points, bars*).
  When one of the 2D types is selected (color or contour mode), the data are displayed in one plot with the channels on the x axis. On the y axis will be plotted the variable selected in the type radio button group: if *Run* is selected the run number will be plotted on the y axis and a new histogram will be created every time the scan variable is changed; if *Script* is selected the scan variable is plotted on the y axis and the bin content is given by the average for all the repetitions; if *Advanced* the advanced variable is plotted on the y axis, the bin content is given by the average for all the positions and a new histogram will be created for all repetions or normal scans; if *position* is selected the position index is plotted on the y axis and a new histogram will be created for all repetions orscans; if *frame* is selected all the frames will be plotted in a time sequence, which is the meaning of the y axis.
  The 2D histogram allows to display only one data argument and is much slower than the 1D histogram, particularly when a large number of channels has to be plotted.
- *Options* allows to change main and axis titles of the plots, the variable on the x axis (only in case of 1D plots) and the scale on the counts axis. It is also possible to select an image file in order to save the plots.

4.6 Settings

The *Settings* tab of the GUI is used to perform opereations that are needed only occasionally and is active only in expert mode i.e. it can be activated by clicking the *Advanced* radiobutton in the *Global settings* box in the *Initialization* tab.

In the *MCS Configuration* box, the settings of the MCS can be seen and can be modified and updated by pressing the button *Change remote settings* (don't forget it!). The remote and trimfile directory are not used up to now.

The *Digital tests* box allows to check the communication with and the functionality of the MCS and the module.

- *Test module*: the digital part of the module is tested and errors are eventually shown.

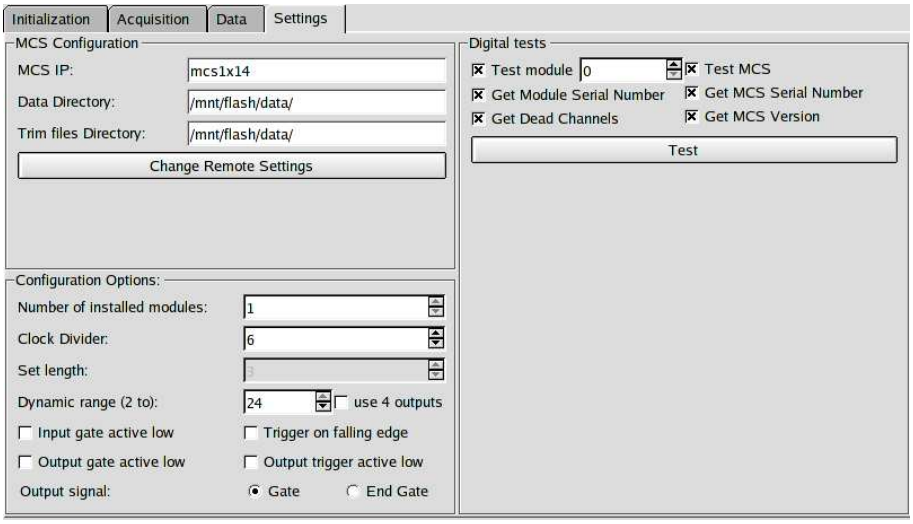

Figure 4.9: Advanced.

- *Get module serial number*: the serial number of the module is returned.
- *Get dead channels*: Not yet implemented: it will always return 0.
- *Test MCS*: The communication with the MCS and the programming of the FPGA are tested. The test takes several minutes since it checks the reliability of the communication between FPGA and embedded linux system. It also checks that the firmware and the software running on the MCS are updated.
- *Get MCS serial number*: Returns the MCS serial number i.e. the last 32 bits of the MAC address of the MCS.
- *Get MCS version*: Returns the MCS FPGA software version.

The *Configuration Options* box allows to set several parameters useful for the acquisition.

- *Number of installed modules* defines the number of modules to be read out in a multimodule system. If you change it after starting the GUI, it might be that not all data are displayed by the plots (but they will all be correctly saved in the files).
- *Clock Divider* Defines the divider of the 100 MHz clock in order to perform the acquisition. It defaults to 6 (17 MHz) clock but in case you have long cables between the MCS and the modules or you experience some errors during the digital test of the module, you can increase it up to 15 until you don't solve these problems.

- *Set lenght* not implemented
- *Dynamic range* allows you to read out a lower number of bits obtaining thus a shorter readout time between frames and a higher frame rate (there are also less data to be transfered). The default is 24 (without 4 outputs), which gives a readout time of 240 μs, otherwise you could select "use 4 outputs" and chose either 24 (240 μs), 16 (190 μs), 8 (125 μs) or 4 (90 μs).
- *Input gate active low* defines the polarity of the external gate. Normally it is active high.
- *Trigger on falling edge* defines whether the acquisition starts on the falling edge of the external trigger signal. Normally it starts on the rising edge.
- *Output gate active low* defines the polarity of the output gate signal. By default it is active high.
- *Output trigger active low* defines the polarity of the output trigger signal. By default it is active high.
- Only in case of MCS1 *Output signal* selects wether the signal coming out of the *out* lemo is the internally generated *gate* or the *end of gate* (trigger).

Chapter 5

Test, calibration and trimming of the detector

Depending on the usage of the system the generation of trim files (e.g. trimming with noise) an energy calibration (conversion from DAC units to energy for the threshold voltage) and/or an angular calibration (channel number to angle conversion for large systems used for powder diffraction) is necessary. The trimming with noise is (and has to be done without X-rays). For the energy and angular calibrations specific dataset have to be taken and then be analyzed. Currently the analysis is done outside the GUI using root macros. Usually the detector is used with the trimbits optimized to give the lowest threshold dispersion.

5.1 Test of the modules

The modules of the detector are usualy shipped with tested and the results are shown in an evaluation sheet as shown in figure 5.1.
Here is a brief description of how the tests are performed.

Module number: the serial number is read out and it is checked that it corresponds to its label;

Digital test: the digital part of the readout electronics is tested by reading out some fixed data sent to the counters and c hecking that they correspond;

Noise trimming: the module is trimmed with noise (see section 5.2.1) using the parameters:

- Settings=standard
- Acquisition time= 100 ms
- Vthreshold=700 DACu

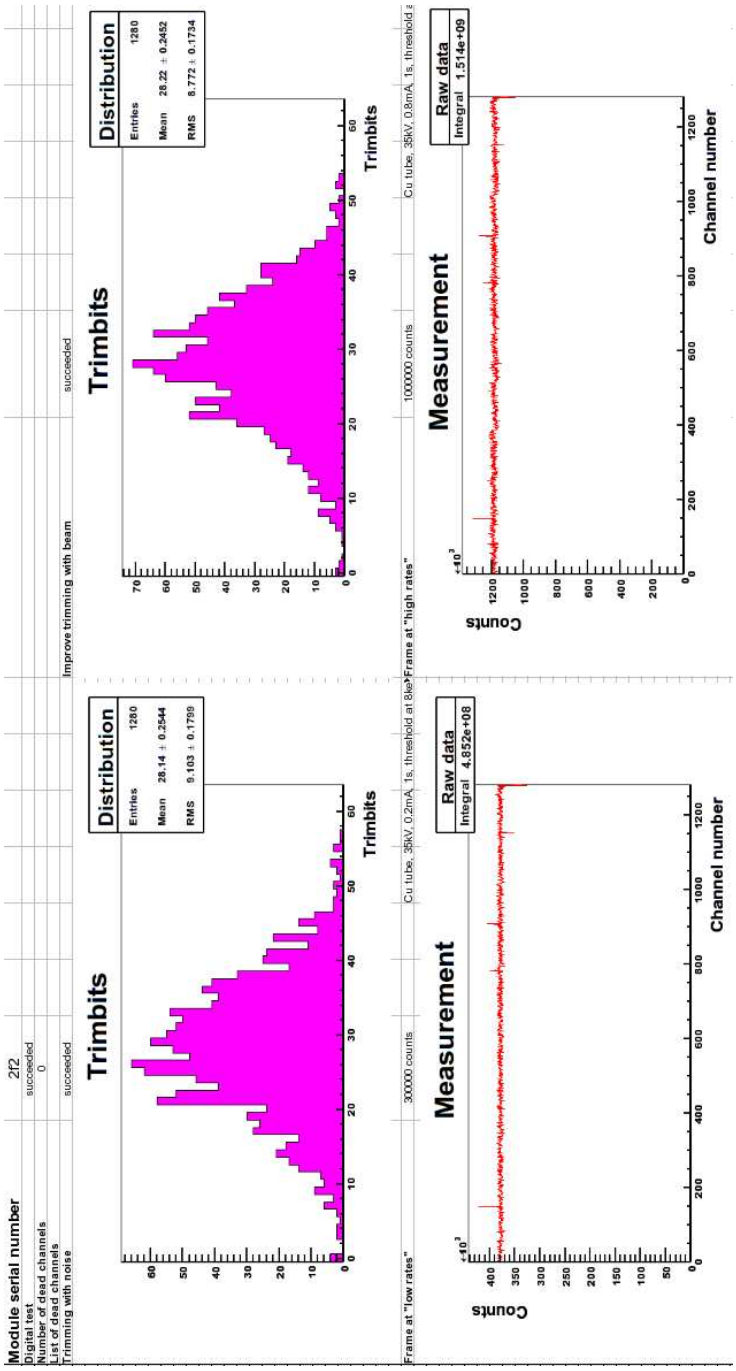

Figure 5.1: Example of a module evaluation sheet.

- Counts/pixel=500
- Resolution=4

The resulting trimbits distribution is shown in the module evaluation sheet (see figure 5.1) and should be gaussian-like, but in particular there should be only few channels with trimbits equal to 0 or 63.

Improve trimming: the module trimming is then improved by using the X-rays. Direct irradiation from a Cu anode X-ray tube set at 35 kV and 0.2 mA is used. The threshold is set at about 5 keV (about 700 DACu), but since the modules are not calibrated in energy (see section **??**) the correct value is found by considering that the module should count about 300 kCounts/strip per second. The trimming is then performed with an acquisition time of 1 s.
The resulting trimbits distribution is shown in the module evaluation sheet (see figure 5.1) and should be gaussian-like, but in particular there should be only few channels with trimbits equal to 0 or 63.

Frame at low rate: in order to evaluate the quality of the module and the number of dead channels, a frame with the module already trimmed is acquired using X-rays from a Cu anode X-ray tube set at 35 kV and 0.2 mA. The data acquired in 1 s are shown in the module evaluation sheet. The channels counting less than half of the average are considered dead. The channels deviating slightly from the average are usually simply badly trimmed and their counts can be corrected using a flat field correction (see figure 5.1).

Frame at high rate: in order to prove that the module can support a counting rate higher than 1 MCounts/strip per second a frame with the Cu anode X-ray tube set at 35 kV and 0.8 mA is acquired. The average number of counts should be about 1 milion counts/strip. The data acquired in 1 s are shown in the module evaluation sheet (see figure 5.1).

5.2 Trimming of the detector

When no trim files for the installed modules exist in the default directories specified in the config file (CAL_TRIMFILE_STANDARD, CAL_TRIMFILE_FAST and CAL_TRIMFILE_HIGHGAIN as explained in section **??**) the default trim files with all trimbits set to 0 are loaded from TRIMFILE_STANDARD, TRIMFILE_FAST and TRIMFILE_HIGHGAIN. The trim files can also be selected with the Browse button and loaded by pressing the load button. If *file name contains sn* is not checked, the specified trimfile will be loaded for all modules (the extension must then be specified), while if the checkbox is not set, the extension should not be specified ".snxxx" will be attached to the specified file name for each module and all the installed modules will load their specific trim file.

5.2.1 Noise trimming

The first step in the trimming procedure is to trim with noise (this is often sufficient). This has to be done for all the settings which are foreseen to be used (highgain, standard and fast).
The procedure for the noise trimming is as follows:

1. In the *Initialization tab* click on the settings for which you want to trim (e.g. standard)

2. In the *Initialization tab* click on the *advanced* radio button to make the trimming accessible.

3. In the *Acquisition tab* set the acquisition time to 100 ms, the repetion to 1 and the delay between frames to 0.

4. For noise trimming usually the default parameters as described in **??** work (and are already loaded when opening the GUI). To verify the threshold setting it is best to make a threshold scan. To do this go to the *Data* tab, in the Data display section select the 2D color and type advanced option. IN the *Acquisition* tab select your data directory. Set the number of positions to 0. Select Scan, Type threshold. Typical values for the range are 500 to 900 with a step size of 10. Then click on the start button to perform the threshold scan. After the threhold scan has finished an image similar to the one in 5.2 should be shown. Depending on the system the number of modules may vary. If the plot is similar to the one in 5.5 the noise trim files did already exist and have been loaded when selecting the settings. In this case you don't need to trim with noise again.
   Set the parameter Vthreshold in the *Trimming* box (*Initialization tab*) is 10-30 DAC units below the onset of the noise for the module with the lowest threshold offset. Since the modules have differences in the offset and gain the onset of the noise varies.
   You can usually leave the remaining parameters unchanged (Counts/pixel=500; Resolution=4).

5. Select the directory where the noise trim files should be written and the filename, to wich will be attached the extension given by the module serial number (.snxxx). If you want the trimfiles to be loaded authomatically when the global settings are selected, select the default directory specified in the config file (CAL_TRIMFILE_STANDARD, CAL_TRIMFILE_FAST and CAL_TRIMFILE_HIGHGAIN as explained in section **??**) . Click on *Trim* to start the noise trimming process. After the trimming has finished lock at the plot and the distribution of the trim bits. The distribution should be around 32±5 and should look gaussian. An example distribution is shown in figure 5.3 and an example plot in 5.4. If the distribution is too much off center change the counts/pixel, if it is too narrow reduce the resolution (set it to 3), if it is too wide increase it (set it to 5). Make sure not too many channels have a trim value of 0 or 63.

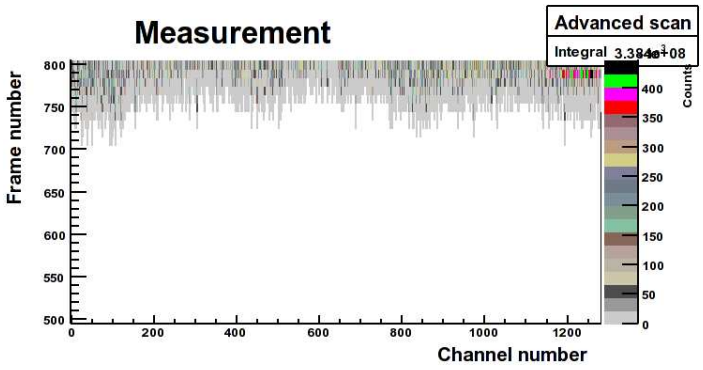

Figure 5.2: The untrimmed threshold scan.

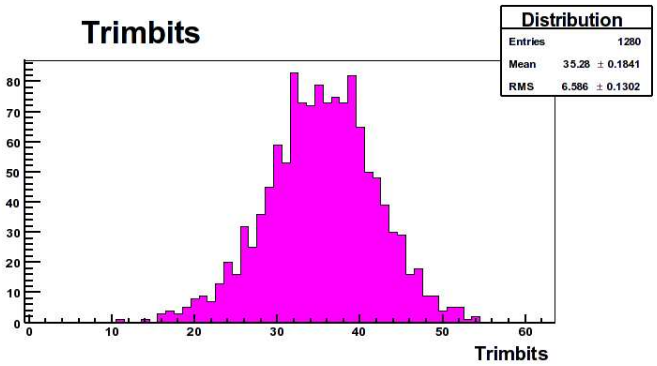

Figure 5.3: The distribution ofd the trimmbits.

6. Execute the treshold scan again to verify the trimming was done properly. A plot similar tho the one in figure 5.5 should appear.

5.2.2 Improve the trimming using X-rays

The improvement of the trimming acquired with noise is not essential: at 12 keV an untrimmed module has a threshold dispersion which is about 1.4 keV and is already reduced to 200 eV at 12 keV by the noise trimming. At lower energies the noise trimming will be more effective, wil the threshold dispesion will be still larger at higher energies. The trimming improvement reduces the threshold dispersion to 140 eV at 12 keV and is expected to be almost constant at all energies. For this reason it is suggested to perform the trimming improvement only when a small threshold dispersion is really important (e.g. to avoid flat field

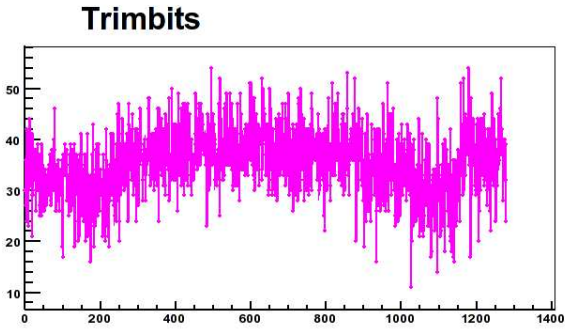

Figure 5.4: The trimbits for all the channels.

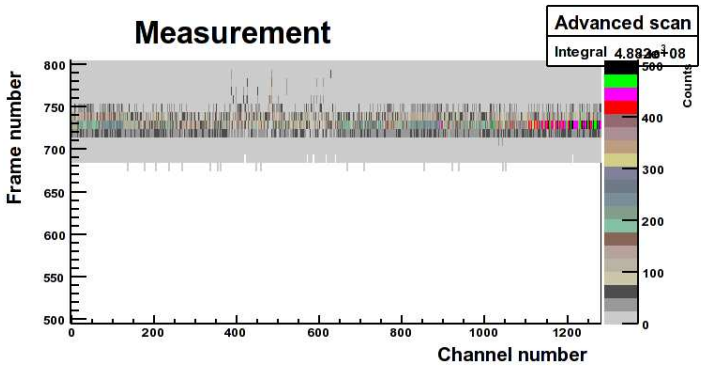

Figure 5.5: The trimmed threshold scan.

corrections) and it will probably be not worthy at lower energies (i.e. threshold lower than 6 keV and X-ray energy lower than 12 keV). The procedure for the trimming improvement is as follows:

1. Select the settings of the detector and load the noise trimming file
2. Set the threshold at half of the X-ray energy (better if the detector has already been calibrated in energy like explained in 5.3)
3. Illuminate the detector with a flat field. This is very important to obtain a good trimming.
4. Select the *acquisition time* in the *acquisition tab* so that there are at least 1000 counts/strip per frame (the more counts, the better trimming). Set the repetions to 1 and the delay between frames to 0.
5. Go to expert mode by clicking on *advanced* in the *initialization tab*, *settings* box
6. In the trimming box select the directory where the noise trim files should be written and the filename, to wich will be attached the extension given by the module serial number (.snxxx).
7. Select the *improve* method *S*tart the trimming

If the trimming is correctly performed and the illumination is flat enough, the same trimming can be used every time you will measure at this same energy. The authomatic loading of energy-specific trim files is not yet implemented.

5.3 Energy calibration of the detector

Since the conversion between the threshold DAC units and energy depends on the gain and offset of the channels the energy calibration has to be done for all settings (high gain, standard and fast). For each setting follow this procedure:

- Select the setting in the *Initialization* tab.
- Enter in expert mode by clicking the *Advanced* radiobutton in the *Global settings* box in the *Initialization* tab.
- If the trimfiles are in the correct location and with the correct name, they should be loaded by default every time you select the corresponding settings in the *global settings* box in the *initialization* tab [1]. If the trim files do not yet exist generate them Trim with noise (suggested settings Acquisition time=100 ms, Vthreshold=700, Counts/pixel=500, Resolution=4

[1] The default name of the calibrated trimfiles is trimbits/beamline/*settings*/noise.snxxx where *settings* is the chosen settings. You can change it in src/qDetector.h and then recompile the acquisition program as described in 3.1.

Name=trimbits/beamline/*settings*/noise where *settings* is the chosen settings - maybe Resolution=5 for fast settings and many modules). Check that the trimbits distribution is nice (close to a gauss curve) and only few channel are 0 or 63.
Make sure that the trimfiles are loaded, but this should be done by default after the trimming.

- Execute a threshold scan of the detector with at least three different energies. The more monochromatic are the X-rays, the better the calibration will be (i.e. scattered X-rays are better than the fluorescent emission).
  The scan should range from where all modules count 0 (estimate 850-20·energy(keV) DAcu) and where all modules start having a lot of noise (usually 800 DACu) with a step of 1 or 2 DACu. The acquisition time should be chosen so that there are at least 1000 counts per strip on the plateau.

- Open the file root/CalAllModules.C for editing. Change the value of the following global variables according to your needs:

  - *nmod* is the number of modules of your system.
  - *nscan* is the number of different threshold scans you acquired.
  - *en* is the array with the energies at which you acquired the scans, in keV.
  - *een* is the array with the errors on the energies at which you acquired the scans, in keV. It is usually small, but can be some hundreds eV in case of dirty fluorescent samples.
  - *fn* is the array containing the location and root file name of your data.
  - *run* is the array containing the run index of your data.
  - *startscan* is the array containing the threshold value at which you started the scans.
  - *stopscan* is the array containing the threshold value at which you finished the scans.
  - *stepscan* is the array containing the threshold step of the scans.
  - *ave* is the array containing the average number of counts per strip on the plateau (it must not be too precise).
  - *sn* is the array containing the list of the serial number of the modules to be calibrated. It is important that the list is in the right order, so that the optput calibration files have the extension .snxxx corresponding to the right module.
  - *of* is the location and root file name of the calibration file. The directory should already exist and the extension .snxxx will be attached to the output file.

- Launch root, which you should have already installed on your linux PC
- Execute the following commands in order to load the macros needed for the calibration:

  ```
  root$ .L root/NewMythenMacros.C+
  root$ .L root/CalAllModules.C+
  ```

  You should get a lot of warnings, but no errors.

- Execute the following command in order to run the calibration:

  ```
  root$ EnCalModules()
  root$
  ```

  Reading and analyzing the data takes some time, but, after a while, a canvas should open where the plots of the median of the counts of every module as a function of the threshold should be shown for each energy, fitted with a modified *erf* function in order to find the inflextion point. The last plot of the canvas should represent the inflexion points as a function of the energies, and by fitting it with a straight line it is possible to calculate the offset and gain for each module i.e. calibrate it as a function of the energy. Please check that this automated fitting procedure succeeds. In case you see many fitting errors you should try to check wether the variable you edited in root/CalAllModules.C are all correct or try to edit the fitting procedures in the two root macro files (sorry!).

- Copy the calibration file you obtained to calibration/*settings*.snxxx [2] By doing this the correct threshold for each module will be calculated every time you change the *threhsold energy* in the *global settings* box in the *initialization* tab, you have loaded some default settings and you are not in expert mode.

5.4 Angular calibration of the detector

This section is specifically (but not only) for larger systems for powder diffraction. It is assumed that the scripts for moving the detector axis have been adopted to the beamline (see chapter 6). For the angular calibration a Si sample is mounted and the detector is moved in steps of 0.1 deg over the 111 peak. Afterwards the recorded data has to be analyzed to produce the angular calibration file. The procedure for the data taking is as follows:

1. Setup the detector trimmed with the correct settings (not important which) an the threshold at half of the X-ray energy.

[2] The default name of the calibration file calibration/*settings*.snxxx where *settings* is the chosen settings. You can change it in src/qDetector.h and then recompile the acquisition program as described in 3.1.

2. Mount the silicon capillary and select the acquisition time to verify that the peak has a height of more than 5000 counts. Make sure the detector is operated in the linear region of the count rate, otherwise use more filters and a longer acquisition time. The energy is not very important, but it should not be very low (¿10 keV).

3. Move the detector to the angular position such that the Si111 peak is just before the first module.

4. Select both header before and header after (the detector position needs to be written in the header files!) and click on *scan*, *before each frame* selecting a scrip that moves the motor. In the scan variable range set the start position such that Si111 peak is just before the first module, the stop position when the Si111 peak is already out of the last module (e.g. start position - (detector range +5)) and a step of 0.1 deg.

5. Start the acquisition. It is very likely that if your detector is large it will take several minutes or even hours.

For the data analysis (will be upgraded):

1. Start root

2. Edit the file $MYTHENDIR/root/convertNewMythenIInew.C to match your setup (epics variable, direction etc.)

3. Load the macro file convertNewMythenIInew.C:

   ```
   root$ .L $MYTHENDIR/root/convertNewMythenIInew.C+
   ```

4. Run the analysis function:

   ```
   root$ fitangle(fname,extension, start, stop, startangle, stopangle);
   ```

   where fname is the file name (with complete path), extension should be ".raw", start is the start position, stop is the stop position, start angle should be the position of the silicon peak -0.5 deg and stop angle the position of the silicon peak +0.5 deg.

5. The procedure will produce a file ang.off containing the angular calibration coefficients for your detector. Copy it to your default location and set it in the config file of the GUI (see section **??**).

Chapter 6

Integration at the beamline

The interfacing of the GUI can easily be done using the scripts described in section

Chapter 7

Offline Data processing

Sometimes it is necessay to process offline the data acquired (e.g. when a new flat field is acquired, a more precise fine offset has been calculated or the data are acquired by using the text client and thus are not processed or several datasets need to be merged together).
For this purpose two different software packages which allow to perform exactly the same actions have been developed, one with and one without the use of root. The one using root allows to display the data and simple C++ programming can be implemented in order to loop over large amounts of data, but the package without root can also be used in case of systems where root is not installed.

7.1 Postprocessing using root

Launch root and load the post processing macro library:

`root$ .L $MYTHENDIR/postprocessing_root/processData.C+`

where `$MYTHENDIR` has to be substituted with your directory name.
The postprocessing then consists basically in two phases: the first is the setup of the postprocessing, when the user must chose the type of processing he wants to perform and set the parameters (e. g. angle conversion file, flat field file, bad channel list) and the second is the postprocessing itself.

Otherwise it is always possible to type the commands in the root command line. Here is a list of the commands with a short explanation:

helpMythen() prints an help regarding the functions (like this description)

setNMod(n) sets the number of modules of your detector (i.e. the amount of data contained in the data to posprocess)

setInFargs(anglemask, channelmask, valuemask, errormask) sets the arguments of the files to be postprocessed: anglemask, channelmask, valuemask and errormask can be 1 or 0 depending if the column of the angle, channel number, counts value and error respectively are (1) or are not (0) present in the files of data to be processed

setOutFargs(int anglemask, int channelmask, int valuemask, int errormask) sets the arguments to be printed in the processed output file: anglemask, channelmask, valuemask and errormask can be 1 or 0 depending if the column of the angle, channel number, counts value and error respectively should (1) or should not (0) be printed in the output file

setFfFargs(int anglemask, int channelmask, int valuemask, int errormask) sets the arguments of the flat field file: anglemask, channelmask, valuemask and errormask can be 1 or 0 depending if the column of the angle, channel number, counts value and error respectively are (1) or are not (0) present in the flat field file

setInFormat(char fstr) selects the format of the counts in the files to be processed. fstr can be 'f' (float) 'd' (integer) 'x' (hexadecimal) 's' (binary)

setOutFormat(char fstr) selects the format of the counts in the processed output file. fstr can be 'f' (float) 'd' (integer) 'x' (hexadecimal) 's' (binary)

setFfFormat(char fstr) selects the format of the counts in the flat field file. fstr can be 'f' (float) 'd' (integer) 'x' (hexadecimal) 's' (binary)

setSum(int minindex, int maxindex)

unsetSum()

setFlatField(char *fffile)

unsetFlatField()

setRateCorrect(float tau)

unsetRateCorrect()

setAngleConversion(char *angcalfile, float globaloffset)

setAngleConversion(float globaloffset)

unsetAngleConversion()

setMerging(int npos, char *angcalfile, float globaloffset)

setMerging(int npos, float globaloffset)

unsetMerging()

setBadChannel(char *badchanfile)

setBadChannel()

unsetBadChannel()

setIcNormalize()

unsetIcNormalize()

setGraph()

unsetGraph()

processData(char *infile, char *ext, char *outfile)

```
Help for the procedures for the MythenII data analysis.
processData(infile, ext, outfile)       processes the data as selected with the other funct
         infile is the path and root of the file.
         If summation or merging must be performed ther should be 1 or two indexes, which wi
         it is assumed that the run index follows the position index, so the file name will
         ext is the extension of the input file (e.g. .dat, .raw, .raw.ff)
         outfile is the complete name of the output file e.g. outpath/outfile.dat - take car

currentSettings()        shows the settings chosen for the post processing
setSum(minindex, maxindex)       sums the runs between minindex and maxindexunsetSum()   wor
setFlatField(fffile)     does the flat field correction using the flat field file fffile and
unsetFlatField()         won't do the flat field correction any more
setRateCorrect(tau)      does the rate correction using the deadtime tau (or the default if
unsetRateCorrection()    won't do the rate correction any more
setAngleConversion(globaloffset)         performs the angle conversion on the files using th
unsetAngleConversion()   won't do the angle conversion any more
setMerging(npos, globaloffset)   performs the angle conversion on the files using the specif
unsetMerging()   won't do the data merging any more
setBadChannel()          won't write the channels contained in the badchannellist
unsetBadChannel()        won't get rid of the bad channels any more
setIcNormalize()         will normalize the counts using the ionization chamber readoutunset
```

It might be convenient to group alle the commands relative to the to phases is two root macros (e.g. setupProcessing.C and runProcessing.C) which can then be executed:

```
   root$ .x setupProcessing.C
root$ .x runProcessing.C
```

which can be particularly useful when the same setup has to be done several time over many root session.

```
  /* infile should be of the type */
  // "run_10"    no merging and no sum
  // "run_%d" or "run_p%d_10" in case di merging or summing, where %d substitutes the index
  // "run_p%d_%d" in case of summing and merging where it is assumed that the first %d is re

  /* ext should be ".raw" */

  /* outfile is the complete path of the output file */
```

```
 int minrun=0;
 int maxrun=1251;
 char *infile1="mydata/silicon_%d";
 char infile[100], outfile[100];
 char *outfile1="mydata/silicon_%d.ff";




for (int irun=minrun; irun<=maxrun; irun=irun+1) {
 sprintf(infile,infile1,irun);
 sprintf(outfile,outfile1,irun);
 processData(infile,".raw",outfile);


}
```

7.2 Postprocessing without root

```
#edit this file to configure your post processing
#please comment out the unwanted features using # at the beginning of the line

#select the number of modules to be processed
nMod=23

#select the bad channel list
badChan=calibration/badchannel_detector.list

#select the angle calibration file
angCal=calibration/ang.off

#select fineOffset - nothing if no angle conversion
fineOffset=0.0

# select format of the input file
#inFormat=angle channel value error
inFormat=channel value

#select format of the flat field file
#ffFormat=angle channel value error
ffFormat=channel value

#select format of the desired output file
#outFormat=angle channel value error
```

```
outFormat=angle value error

#select the directory to read/write to (also the flat field)
dir=/scratch/data

#select flat field file - nothing if not necessary
# use path relative to dir
ffFile=ff_1.raw

# select the extension of the input files
ext=.raw


# select number of positions to be merged (probably does not work for more than 8) and the p
nPositions=2

################################
# If your file names are similar to the ones given by the GUI
################################

# File name, where %d substitues the position index and the run number (only one index will
inFiles=run_p%d_%d
outFile=run_%d

# minimum run index when you what to loop over many runs
minRun=0

#maximum run index when you want to loop over many runs
maxRun=10



################################
#In case it is not simple to write a file format,write list of files: they will be merged in
################################

#the first of each group will be the outputfile, then the posiitons

filelist=

silicon.dat
silicon_p1_1
silicon_p2_1
silicon_p1_2
silicon_p3_2
```

```
run.dat
run_p1_0
run_p3_1
run_4
run2_0
```